Preprocessing of Configuration Space for Improved Sampling Based Path Planning

Titas Bera[a,1], M. Seetharama Bhat[a], D. Ghose[a]

[a]Department of Aerospace Engineering, Indian Institute of Science, Bangalore

Abstract: Sampling based planners have been successful in path planning of robots with many degrees of freedom, but still remains ineffective when the configuration space has a narrow passage. This paper presents two new techniques of preprocessing the configuration space. The first technique called a Random Walk to Surface (RWS), uses a random walk strategy to generate samples in narrow regions quickly, thus improving efficiency of Probabilistic Roadmap (PRM) based planners. The algorithm substantially reduces instances of collision checking and thereby decreases computational time. The method is powerful even for cases where the structure of the narrow passage is not known *a priori*, thus giving significant improvement over other known methods. The second method, by preprocessing the configuration space, improves the efficiency of Rapidly Exploring Random Tree (RRT) like planners by identifying key regions of the configuration space to search for a solution path. The Experiments show a significant improvement in efficiency for both PRM and RRT like planners.

Key words: Robot Motion Planning, Randomized Algorithm, PRM, RRT

1. INTRODUCTION

Several areas beyond classical robotics, are benefiting from advances in algorithmic motion planning. Examples include structural studies in biology, computer games, path planning of unmanned aerial vehicles among the others. The success of these algorithms on different instances is because of availability of an abstract formulation of the model of many different real world applications, and ability to solve difficult problems in a reasonable time. Modern motion planning begins with the introduction of the notion of configuration space by Lozano-Perez and Wesley [1], [2]. In the configuration space the robot is reduced to a point; hence the motion planning problem becomes that of finding a path from an initial point to a goal point in the configuration space. However, the explicit mapping from workspace obstacles to configuration space is in general difficult. Early studies showed that the basic version of this problem is PSPACE-complete [3], [4] [5] and the best exact deterministic algorithm known is exponential in the dimension of the configuration space. On the other hand, real world problems generate instances with high dimensional configuration spaces.

Email addresses: titasbera@aero.iisc.ernet.in (Titas Bera), msbdcl@aero.iisc.ernet.in (M. Seetharama Bhat), dghose@aero.iisc.ernet.in (D. Ghose)

URL: http://www.aero.iisc.ernet.in (Titas Bera)

[1]Corresponding author

Since the mid nineties, in order to break this "curse of dimensionality", sampling based approaches were introduced. The approach is based on the generation of samples to acquire information about the problem instance being solved. The first generation of these algorithms heavily relied on random samples. In contrast, there has been a recent trend to use deterministic sampling schemas [6]. For motion planning problems, samples are stored in a data structure which represents an approximation of the configuration space, as opposed to its exact combinatorial representation. The data structure is usually composed of nodes, that is, samples in the configuration space, and links, that is, valid paths connecting samples. Nodes and links can be stored in the form of graphs or trees. The entire continuum of configuration space is then approximated to a network of nodes. The implementation of these algorithms is usually quite simple. The price to pay is completeness.

Traditional combinatorial motion planning algorithms are complete, that is, they will find a solution if one exists, and will report failure otherwise. Algorithms based on randomly generated samples aims for probabilistic completeness. This means that if a solution exists, the probability to find it converges to 1 when the computation time approaches infinity [7]. Despite probabilistic completeness, randomized algorithms such as the Randomized Potential Field Planner (RPP) [8], the Probabilistic Road-map (PRM) family [9], Rapidly-

Exploring Random Trees (RRTs) [10], and others, have dominated the field of motion planning for over a decades.

Despite the success of sampling based path planners, motion planning in high dimensional configuration space is difficult. Several instances of the problem have been even proven to be undecidable. It is unlikely that sampling, can overcome such difficulty entirely. For PRM like planners one such difficulty arises when configuration space posses narrow passages. Narrow passages are those critical regions in the configuration space that require a number of sample points in order to capture the free space connectivity. This difficulty posed by narrow passages and its importance were noted in early work on PRM planners [9]. Several sophisticated sampling strategies can remove this difficulty to a large extent, but a satisfactory answer remains elusive. Likewise, the issue of exploration vs. biasing issues in case of RRT poses a significant challenge. The 'rapid' nature of RRT is due to the fact that the samples pulls the tree towards the unexplored areas of configuration space. This leads to a creation of large number of unnecessary nodes before obtaining a final solution. The exploration weightage and biasing weightage toward the goal is a critical design issue.

In this paper we propose a new methods which is suitable to tackle problem of narrow passages in high dimensions for PRM. We also present a sequential sampling scheme which can guide an RRT effectively toward the goal state. In Section 2 we formulate the general motion planning problem. In Section 3 we briefly explain PRM method, problem of narrow passage and related work. Section 4 describes our algorithm for narrow passage sampling. Section 5 discuss the implementation issues and result. Section 6 reports RRT algorithm and related exploration vs. biasing issues. Section 7 gives the details of our proposed methodology to improved performance of RRT, while in subsequent sections we discuss and summarize the implementation issues and results.

2. PROBLEM FORMULATION

The configuration of a robot with n DOFs can be represented as a point in an n-dimensional space , called the configuration space C, which is locally like the $d-$Dimensional Euclidian Space $\mathbb{R}^d$. A configuration q is free if the robot placed at q does not collide with the obstacles or with itself. We define the free space C_{free} to be the set of all free configurations in C. There are a finite number of obstacles in the configuration space, which are closed bounded sets O_i, $i = 1, 2, \cdots, m$ and we can fairly assume that they are pairwise disjoint. Let the starting configuration be $x_{start} \in C_{free}$ and the final configuration be $x_{goal} \in C_{free}$. For convergence issues we define a rather general goal subset X_{goal}, than a specific point. Clearly $x_{free} \in X_{goal}$. The motion planning problem is to find a path connecting x_{start} with X_{goal}, that is, a continuous function $f : [0, 1] \rightarrow C_{free}$ such that $f(0) = x_{start}$ and $f(1) \in X_{goal}$. We assume an appropriate collision checker function does exist for instance of the problem. In related literature the configuration space C is frequently substituted with the state space $\mathcal{X}$, which includes both the degrees of freedom and their derivatives. The constraints can be nonholonomic and differential.

3. PRM AND NARROW PASSAGE PROBLEM

Probabilistic road-map methods solves motion planning problems that do not involve dynamics of the robot or have negligible dynamics. A classic multi-query PRM planner proceeds in two stages. In the first stage, it randomly chooses samples from C_{free}, called milestones according to a sampling scheme(which in general uniform). It then uses these milestones as nodes to construct a graph, called a road-map, by adding an edge between every k pair of milestones that can be connected via a simple collision-free path, typically, a straight-line segment. After the road-map has been constructed, multiple queries can be answered quickly in the second stage. Each query consists of an initial configuration and a goal configuration, and asks for a collision-free path connecting x_{start} and x_{goal}. The planner first finds two milestones in the road-map, such that x_{start} and x_{goal} can be connected to these nodes, respectively. The rest of the job is to search for a path (Or may be search for the shortest path).

If the configuration space possess a narrow passage, then to capture the connectivity of C_{free}, it is essential to sample milestones in narrow passages. This, however, is difficult, because of small volumes of narrow passages. Any volume-based sampling distribution is likely to fail. Uniform distribution may not work well when the dispersion of the samples is higher than the narrow passage volumes. Furthermore, when dealing with multiple degree of freedom robots, we do not have an explicit representation of C_{free} and cannot locate narrow passages directly by processing the global geometry of C_{free}. Intuitively, one can sample more densely near obstacle boundaries because points in narrow passages lie close to obstacles. This method called Gaussian sampler [11] is a simple and efficient algorithm that uses this idea. However, in some cases, many points near the obstacle boundaries lie far away from narrow passages and do not help in improving the connectivity of road-maps. So, despite the improvements, sampling near obstacle boundaries may generate many samples in uninteresting regions. Other geometric approaches [12], [13], [14] also exist but those are expensive to implement in high-dimensional configuration spaces.

Perhaps the most appealing scheme to answer the problem of narrow passage is the Bridge test [15]. Here when a sample lies within an obstacle, one uses this information to build a bridge whose two end points lie in the obstacle while the mid point lies in C_{free}. This method, although it requires a high computational time (because of more number of collision checking), can be very effective. In the following, we give the standard bridge test algorithm (Randomized Bridge Builder).

Algorithm: **R**andomized **B**ridge **B**uilder (RBB)

For $i \leftarrow 1$ to K
$q_1 \leftarrow$ *Random Configuration*(Uniform Distribution)
If *Clearance* (q_1) returns False
then $q_2 \leftarrow$ *Random Configuration*(λ_x)
If *Clearance* (q_2) returns False
then $q_3 \leftarrow$ mid point of $\overline{q_1 q_2}$
If *Clearance*(q_3 returns True
then Insert q_3 to List G as a new milestone
Return(G)

Here "Clearance" is a collision checker which returns *true* if the sampled configuration is in C_{free} and λ_x is a multi-dimensional Gaussian distribution with a pre-specified variance σ. The performance of this algorithm depends heavily on the choice of σ. If one chooses σ to be very small it takes a very long time and numerous collision checking before to come up with a point in C_{free}. Also a large σ results in large number of redundant configurations. The computation time also increases with the increase in number of obstacles.

4. RANDOM WALK TOWARD SURFACE

Next we present our algorithm, which generates the sample points in the surface of the configuration space obstacle. The idea is, since in high dimension one does not know the suitable value of σ, and there is no a priori knowledge about narrow passage geometry, therefore the most idealistic approach will be to generate points on the surface of the obstacle. Note that in the Gaussian sampling scheme, one tries to generate points on the surface of the obstacles. But again a unsuitable value of σ for a particular problem instance may take a very long time to generate a point in the narrow passage. Our algorithm is based on a simple philosophy, such as, once a sample point is generated within the obstacle one can perform a discrete random walk with a fixed length size a, until the particle comes out of the obstacle.

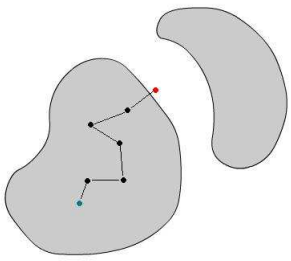


Figure 1: RWS illustration. Shaded regions are obstacles

Algorithm: **R**andom **W**alk Towards **S**urface (RWS)

For $i \leftarrow 1$ to K
$q \leftarrow$ *Random Configuration*
If $q \in C_{free}$
then add to List L
else Flag $\leftarrow 1$
while Flag = 1
$\triangleright$ Select Random Direction
$q' \leftarrow$ *Random Walk* (q, p, a)
If $q' \in C_{free}$
then add q' to the list L
Flag = 0
else $q \leftarrow q'$
Return L

The most important aspect of the algorithm is the reduction in collision checking computations. Both in RBB and Gaussian sampling strategy, to generate a point inside the obstacle, one has to check if the sampled random configuration belongs to either of m, O_i's. This increases with increase in number of obstacles. In our algorithm, once a point is generated inside obstacle O_i', it has to only check if the point belongs to only that obstacle O_i' or not. This reduces computation time drastically, as we shown on Table 1.

As in Gaussian sampling strategy, our algorithm generates unnecessary points on the surface of the obstacle which may be far away from critical regions. We accept this as a disadvantage of the algorithm, and propose a discrimination method to eliminate unnecessary points.

We define a heuristic distance $d(x, y) \propto \frac{1}{N}$ as threshold, where N is the number of points. Once all the points are generated, for every point $p \in N$ a neighbor list is created which contains points that lie within $d(x, y)$. If all the neighbors originate from the same obstacle then p is deleted.Thus following algorithm describes the process

Algorithm: **E**limination(L)

$N \leftarrow Size(L)$
For $i \leftarrow 1$ to N
$Q \leftarrow$ *Neighbors* ($q(i), d$)
$I \leftarrow q(i).index$
$M \leftarrow Size(Q)$
For $k \leftarrow 1$ to M
if $I = Q(k).index$
then $p \leftarrow p + 1$
If $p = M$
then *delete* $q(i)$
else insert $q(i)$ to List S
Return S

This eliminates unnecessary points. We also try to provide an approximate analysis of RWS in terms of mean escape time τ_{es} required to come out of the obstacle. Although our proposed random walk is discrete, but we approximate this with a continuous time diffusion process. This allows us to

state a much more simple expression for mean escape time τ_{es}. Since the expression for mean exit time is available in existing literature, therefore we only state the result. For a somewhat elementary proof, see Appendix. A more general discussion and in depth analysis of diffusion process can be found in [16]. In particular, the expected exit time from a ball of radius R in d dimension is given by

$$\tau = \frac{R^2 - |x|^2}{d}, \quad x \in \mathbb{R}^d \tag{1}$$

where x is the location of initial position of particle from origin. Therefore, maximum of expected exit time is $\tau_{max} = \frac{R^2}{d}$.

Now the question is, how many steps will be taken up to and including time τ_{max}. We assume the waiting time between successive steps is finite and constant. We denote this by β. Therefore, the expected number of collision checking required before a particle comes out of domain Ω is,

$$N_{col} \sim \frac{R^2}{d\beta} \tag{2}$$

Interestingly this indicates that as the dimension increases it becomes less and less probable for a random walker to return to the starting point as shown by Pólya [17]. This means particles once sampled within the obstacles, are more probable to quickly appear onto the obstacle surface as the dimension goes high.

Instance A	RBB		*RWS* after Elimination
No of Points	σ	Time (Sec)	Time (Sec)
30	2	91.45	11.7247
Instance B	RBB		*RWS* after Elimination
11	2	328.359	9.4742
	5	22.6167	
	10	8.2416	
	20	17.62	
Instance C	RBB		RWS
14	2	165.862	50.21331
	5	26.52	
	10	15.03	
	20	3.144	
Instance D	RBB		RWS
3	2	231.517	9.823
	5	29.02	
	10	17.124	
Instance E	RBB		RWS
3	2	50.50	9.823
	5	5.281	
	10	1.27	

Table 1: Comparison between RWS and RBB

5. COMPARISON RESULTS BETWEEN RWS AND RBB

We compare the samples generated in the critical region and time to generate them for both RBB and RWS algorithms in different critical environments, Fig 3 to Fig 8. We choose several benchmark problem, tested in a Pentium dual core processor and standard MATLAB runtime environment. Configuration space is $[0, 100] \times [0, 100]$. In case of RWS the random walk has a fixed step size of 1 m. The comparison results in Table 1 shows the capability of RWS algorithm compared to RBB. Note that with increase of σ, RBB may generate samples not in the critical region as shown in Fig 7. The differences in the computation time with different σ clearly indicates the importance of proper σ selection. With increase in number of obstacles, computations become slower in case of RBB as shown in Fig 7 (which also evident from Table 1).

6. RAPIDLY EXPLORING RANDOM TREE

Rapidly Exploring Random Tree (RRT) have been shown to be very effective in solving robot motion planning problems in complex configuration space with kynodynamic constraints. RRT was introduced in [18],[10] as an efficient data structure and sampling scheme to quickly search high dimensional spaces that have algebraic constraints (arising from obstacle) and differential constraints (arising from nonholonomy and dynamics). The algorithm incrementally builds a tree whose nodes are different configurations of the robot/vehicle. The edges of the graph correspond to the feasible path between the configurations. The key idea of RRT is to bias the exploration toward unexplored portions of the space by sampling configurations, and incrementally pulling the search tree toward the unexplored portion [19]. In the following we present the basic RRT algorithm.

Algorithm: Build-**R**RT(q_{init})

$T \cdot init(q_{init})$ ▷ Initialize tree T
For $k \leftarrow 1$ to K do
  $q_{rand} \leftarrow$ *Random Configuration*
  Extend (T, q_{rand})
Return T

Extend (T, q)

$q_{near} \leftarrow Nearest\_Neighbor\ (q, T)$
$q_{new} \leftarrow New\ State\ (q_{near}, u)$
If *Clearance* (q_{new}) is True
 then
  $T \cdot add\_vertex(x_{new})$
  $T \cdot add\_edge(x_{near}, x_{new}, u_{new})$
  If $x_{new} = x$
   then Return *Reached*

else
Return *Advanced*
Return *Trapped*

In the *extend* function, a nearest vertex in the tree to a given random sample state, is selected for future expansion. The function New−State makes a motion towards x by applying an input $u \in U$ to x_{new} for some time increment Δt. This input can be chosen at random, or selected by trying all possible inputs and choosing the one which yields a new state as close as possible to the sample [10].

7. PREPROCESSING THE CONFIGURATION SPACE

Many variants and modifications of the basic RRT algorithm can be found in [20], [21], [22], [23] etc, in order to address exploration vs biasing issues, nearest-neighborhood queries, and probabilistic convergence. In [24], it is mentioned that different choice of metric in C_{free} can lead to a drastic difference in the performance of RRT. Apart from this issues one of the biggest problem is the issue of exploration vs biasing. While many used an artificial bias in generating samples towards the goal [25], it is certainly not the best approach because of the potential problem of local minima. One variant is to pick up milestones in Probabilistic Roadmap method and try to push RRT to sequentially follow the PRM vertices [10]. But in general, for dynamical systems, the problem of reachability may occur, and one never knows whether a desired milestone is reachable or not *a priori*. In [20], [21] an adaptive biasing is used. There the focus is on the selection of the best among k−nearest neighbors, and on adaptively increase or decrease the sampling bias. But again this approach depends on too many tuning parameters that depend on configuration space criticality.

The RRT algorithm has a known exponential bound on its run length tail probabilities. Its nearest-neighbor operation implies that individual iteration require increasing time. Therefore, it is reasonable to assume that run time tail probabilities may arise that are heavier than exponential. It is shown in [26], that two independent forward search RRT, solving the same query, can reach roughly the same level of progress at drastically different run lengths.

So far there is little proof on probabilistic completeness of RRT for arbitrary kino-dynamic problem. In [10], a theorem states that for both convex and non-convex state space and for holonomic path planing problem, the probability, that the dispersion of the RRT vertices will be less than an arbitrary positive threshold, will converge to 1 as the number of vertices approaches infinity.

Theorem [10]: *Suppose x_{init} and x_{goal} lie in the same connected components of a convex/non-convex, bounded open n dimensional state space. The probability that an RRT constructed from x_{init} will find a path to x_{goal} approaches one as the number of RRT vertices approaches infinity.*

Based on the above theorem, we realize that, identification of necessary segments of the configuration space (state space) that is worth searching sequentially for the goal, may generate a lower dispersion value for a given number of RRT vertices, that is, the solution can be found 'quickly'. This motivates us to pre-process the configuration space to find most suitable regions to be sequentially explored before becoming close enough to x_{goal}. As an analogy, imagine RRT to be a big ship approaching a harbor and requiring the guidance of a small pilot ship which knows the different channels to approach having different quality measures. An RRT used for kinodynamic problem therefore requires a set of guiding milestones representing the key segments. These guiding milestones cannot be chosen using elementary PRM, because those may not be reachable from the current configuration. Before describing our approach, we define the following:

1. State Space: A topological space $X \in [0,1]^n$.
2. Boundary Value: $x_{init} \in X$ and $X_{goal} \subset X$.
3. Collision Detector: A function, $D : X \rightarrow \{true, false\}$ that determines whether global constraints are satisfied for state x.
4. Inputs: A set U which specifies the complete set of controls.
5. Incremental Simulator $A(\eta\Delta t)$: Given the current state $x(t)$ and inputs applied over a time interval $\eta\Delta t$, the incremental simulator computes $x(t + \eta\Delta t)$.
6. Metric: A real valued function $\rho : X \times X \rightarrow [0, \infty]$.
7. Reachable set:
$R(x^k, t^k, \eta\Delta t) = \{x \in X \mid u \in U, x = x^k + \int_{t^k}^{t^k+\eta\Delta t} f(x,u)dt$
8. Voronoi partition of set P: Let $P := \{p_1, p_2, ...p_n\}$ be a set of n distinct points in $\mathbb{R}^n$; these points are the sites. The voronoi diagram of P as the subdivision of the space into n cells, one for each site in P, with the property that a point q lies in the cell corresponding to a site p_i if an only if $\rho(q, p_i) < \rho(q, p_j)$ for each $p_j \in P$ with $j \neq i$. The voronoi partition of set P is $V(P)$, Fig(2).

We assume no *a priori* knowledge about the configuration space (state space). By uniform sampling scheme we choose N number of points in the free portions of the state space. The goal is to generate a non-uniform partition of the space. The suitable value of N of course depends on the dimension and range of the problem instance. Next, we do a Voronoi partition of the space based on these N registration points (sites), $V(P)$, according to the given metric ρ. The goal is to find a single step transition probability of a particle from region i to region j of the Voronoi partition. One problem may occur that, once we partitioned the state space, connectivity of a region or cell may be lost. Since the shape of the configuration space obstacle is unknown, the lowest grid resolution required to overcome the problem is also unknown. Only we can assume that the C−space obstacles O_i's having large measure are more likely to create problems for a specific N. It is clear that points on the surface of the obstacle generated in an uniform manner can reduce the problem of disconnectedness

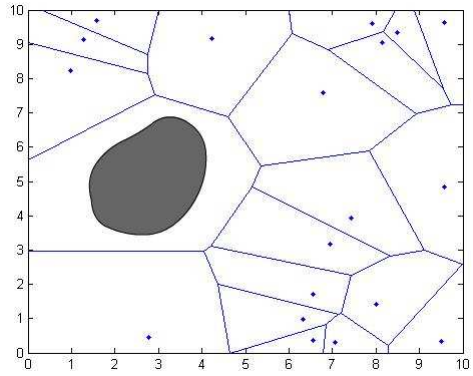


Figure 2: Voronoi Partition of space

substantially. We use therefore, the RWS algorithm to find points on the surface of the obstacles. We call this as enrichment of the initial Voronoi cell distribution. The degree of reduction is dependent on the number M of additional points generated.

We do a repartitioning using Voronoi grid of $N+M$ points. Next, using Monte-Carlo method we approximate the volume measure of the free portions of Voronoi grid cells, with a confidence level of 0.95 [27]. We generate N' particles in C_{free} by uniform distribution.

Once we have an approximate measure of the free volume in each cell w_i, we give random excitation input to the individual particle for a time period of $\eta\Delta t$. The incremental simulator gives us the position of the particle within a Voronoi cell. We then calculate the transition probability P_{ij}, defined as the probability that the particle at cell i at time t, will go to cell j at time $t+\eta\Delta t$. For all such grid cell we get the η step transition probability matrix P_η. Assuming the time interval $T=\eta\Delta t$, we rename this P_η as single step transition probability P_{ij}. Note that for the transition probability calculation, the underlying assumption of Markovian nature of the process exists. So

$$P_{ij} = \frac{\text{Number of particles that goes into cell } j \text{ from cell } i}{\text{Number of particles fall inside cell } i}$$

$$\sum_{i=1}^{N+M} P_{ij} = 1 \quad \forall j \in [1,2,...,N+M]$$

Once we have the transition probability of the entire grid regions we can calculate, such as, 1^{st} passage times from the $Cell_i$ containing the initial configuration to $Cell_g$ containing final configuration and most probable path γ from $Cell_i$ to $Cell_g$. Assuming one such path exists, the most probable path γ_m has the maximum probability over all such paths.

Next, we describe the dynamic programming procedure to find γ_m for a given $V(P_{N+M})$.

Algorithm for Most Probable Path: To find the single best cell sequence $Q=\{q_1,q_2,...,q_T\}$ ($q_1,q_2,...,q_T$ are corresponding cells) we need to define the quantity

$$\delta_t(i) = \max_{q_1,q_2,...,q_{t-1}} P[q_1,q_2,...,q_t=i] \tag{3}$$

that is, $\delta_t(i)$ is the best score along a single path at stage t which ends at cell S_i. By induction

$$\delta_{t+1}(j) = \max_i \delta_t(i) P_{ij} \tag{4}$$

To retrieve the cell sequence, we need to keep track of the argument that maximized (4), for each t and j. We do this via the array $\psi(j)$. The complete procedure for finding the best state sequence can now be stated as

1. Initialization

$$\delta_1(i) = \pi_i w_i \tag{5}$$
$$\psi_1(i) = 0 \tag{6}$$

2. Recursion

$$\delta_t(j) = \max_{1\le j\le N} \delta_{t-1}(i)\, a_{ij} w_i \tag{7}$$
$$\psi_t(j) = \arg\max_{1\le i\le N} \delta_{t-1}(i)\, a_{ij} w_i \tag{8}$$

3. Termination

$$P^* = \max_{1\le i\le N} \delta_T(i) \tag{9}$$
$$q_T^* = \arg\max_{1\le i\le N} \delta_T(i) \tag{10}$$

4. Back Tracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1, T-2, ..., 1 \tag{11}$$

It is important to choose the number of stages T over which the maximization requires to be done. One can evaluate first passage characteristic from cell i to cell j and compute the mean first passage time. There are standard Monte Carlo based procedures that exist for finding approximate mean first passage time T [27]. The interpretation of the most probable sequence cell is that these zones are to be sequentially searched by RRT. The Voronoi registration points of these cells are the guiding milestones.

8. IMPLEMENTATION ISSUES AND RESULTS

Although there is no guarantee that every time the above pre- processing of configuration space lead to the best sequence path (zone). The procedure clearly depends on

1. The non-uniform grid distribution.
2. The number of grid cells.
3. The number of points generated on the surface of the obstacles.
4. The criticality of the configuration space.
5. The reachable set R^k.

While implementing one should carefully evaluate the transition probability matrix, as it is the heart of the whole procedure. Depending on grid distribution one may find cases where $P_{ij} = 1$ which indicates a transient region. There one may generate additional points with increased $\eta\Delta t$ and try to approximate P_{ij}. Also, there may be cases where $P_{ij} = 0$ because of small cell measure. In this case one can excite the corresponding cell registration point with random control input and thus approximate the value of P_{ij}. Depending on the configuration space criticality there may be large uncertainty in evaluating transition probability matrix. Inclusion of uncertainty in P and generating a robust procedure for the method is our future concern. We have tested our method for path planning of nonholonomic mobile robots, although it can be applied to any other configuration space as well. The vehicle dynamics is defined by

$$\dot{x} = v\cos(\theta) \quad (12)$$

$$\dot{y} = v\sin(\theta) \quad (13)$$

$$\dot{\theta} = (v/L)\tan(\theta) \quad (14)$$

The configuration space is $\mathbb{R}^2 \times S^1$. We intentionally created 6 critical obstacle traps so that a conventional RRT requires much longer time to come to the goal region. For each of the 6 problem instances a most probable path is found by our algorithm. Once the sequential milestones are available one can guide the RRT using biased sampling towards the milestones. We used 3000-5000 samples to approximate the cell free volume and to calculate the transition probability. For problem instances A to E (Fig [9-13]) the start configuration is (50,50,0) and goal configuration is (100,100,20) and for problem instance F in Fig 14, the start configuration is (0,0,0) while the goal configuration is (100,100,0). A limited number (usually 10-100, depending on the problem instance) of initial Voronoi cells (N_{Vor}) are created. Enrichment N_η depends on the criticality of the problem instance. One can argue about the added computation required for this approach, but clearly the results shows it is worth investing in the extreme case.

9. CONCLUSIONS AND FUTURE WORK

We developed configuration space preprocessing techniques to improve the efficiency of sampling based planners like PRM and RRT in critical situation. The approaches are promising. In future our focus will be on forming a robust approach to tackle uncertainty in transition probability matrix calculation and consequently development of an improved pre-processing technique.

ACKNOWLEDGEMENT

The first author would like to thank Dr. S. K. Iyer for giving important guidelines on random walks. This research work is partly sponsored by UK-India Education and Research Initiative (UKIERI) programe on " Towards Reliable Smart Adaptable Air Vehicles".

REFERENCES

[1] T. Lozano Perez. Spatial planning: A configuration space approach. *IEEE Transaction on Computers*, 32(2):108–120, 1983.

[2] T. Lozano Perez and M. A. Wesley. An algorithm for planning collision free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[3] J. T. Schwartz and M. Sharir. On the piano mover's problem. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[4] J. Canny. Some algebraic and geometrical computations in pspace. In *Proc. of FOCS*, pages 461–467, 1988.

[5] J. H. Reif. Complexity of the mover's problem and generalization. In *Proceedings of FOCS*, pages 421–427, 1979.

[6] S.R. Linderman and S.M. Lavalle. Current issues in sampling based motion planning. In *Proc in 8th Int. symposium on Robotics Research*. Springer Verlag, 2004.

[7] H. Choset et al. *Principles of Robot Motion: Theory Algorithms and Implementation*. MIT Press, Cambridge, Massachusetts, 2005.

[8] J. Barraquannd and J. C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotic Research*, 10(6):628–649, 1991.

[9] L.E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmap for path planning in high dimensional configuration spaces. *IEEE Transaction on Robotics and Automation*, 12(4):566–580, 1996.

[10] S.M. Lavalle and J.J. Knuffer. Rapidly exploring random tree: Progress and prospects. In B. Donald et al, editor, *Algorithmic and Computational Robotics, New Directions*, pages 45–59. A K Peters, 2001.

[11] V. Boor, M.H. Overmars, and A.F.van der. Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 1018–1023, 1999.

[12] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. In *Proc. 3rd Workshop Algorithmic Found. Robot*, pages 155–168, 1998.

[13] D. Hsu, L. E. Kavraki, J. C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proc. 3rd Workshop Algorithmic Found. Robot*, pages 141–153, 1998.

[14] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space. In *Proceedings of 15th Annual ACM Symposium on Computational Geometry*, pages 173–180, 1999.

[15] S. Zheng, D. Hsu, J. Tingting, H. Kurniawati, and J. H. Reif. Narraow passage sampling for probabilistic roadmap planning. *IEEE Transaction on Robotics*, 21(5):1105–1115, 2005.

[16] I. Karatazas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer-Verlag, Berlin, 1988.

[17] B. D. Hughes. *Random Walk and Random Environments: Vol 1*. Clearson Press, Oxford, 1995.

[18] S. M. Lavalle. Rapidly exploring random tree. In *Technical Report TR 98 -11*, Computer Science Department, Iowa State University, October 1998.

[19] S. M. Lavalle. *Planning Algorithms*. Cambridge Press, 2001.

[20] J. Kim, J.M. Esposito, and V. Kumar. Sampling based algorithm for testing and validating robot controllers. *International Journal of Robotic Research*, 25(12):1257–1272, 2006.

[21] C. Urmson and R. Simmons. Approaches for heuristically biasing rrt growth. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.

[22] E. Frazzoli, M. A. Dahleh, and E. Feron. Real time motion planning for agile autonomous vehicles. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Denver, August 2000.

[23] R. Pepy and A. Lambert. Safe path planning in an uncertain-configuration space using rrt. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006.

[24] P. Cheng and S.M. Lavalle. Reducing metric sensitivity in randomized trajectory design. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 43–48, 2004.

[25] S.M. Lavalle and J.J. Knuffer. Randomized kinodynamic planning. *International Journal of Robotic Research*, 20(5):378–400, 2001.

[26] N. A. Wedge and Michecl S. Branicky. On heavy tailed runtimes and restarts in rapidly-exploring random trees. In *AAAI Workshop*, 2008.
[27] G. F. Fishman. *Monte Carlo:Concepts, Algorithms, and Applications*. Springer-Verkag, Berlin, 1996.

APPENDIX

We begin by assuming a random sample generated within a obstacle and calculate maximum of mean exit time $\max(\tau_{es})$. This is equivalent to evaluate the maximum mean exit time from a domain Ω, of a particle, which is showing brownian motion in a stationary fluid medium. If $b(t,x) \in \mathbb{R}^3$ is the velocity of the fluid at the point x at time t, then a reasonable mathematical model for the position X_t of the particle at time t would be a stochastic differential equation of the form

$$\frac{dX_t}{dt} = b(t, X_t) + \sigma(t, X_t)W_t \tag{15}$$

where, $W_t \in \mathbb{R}^3$ is a white noise. The It$\hat{\mathbf{o}}$ interpretation of this equation is

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dB_t \tag{16}$$

where B_t is $m-$dimensional Brownian motion, $X_t \in \mathbb{R}^d$, $b(t, X_t) \in \mathbb{R}^d$, $\sigma(t, X_t) \in \mathbb{R}^{d\times m}$. We call b the drift coefficient and σ the diffusion coefficient.

We can associate a second order partial differential operator A to an It$\hat{\mathbf{o}}$ diffusion X_t. The basic connection between A and X_t is that A is the generator of the process X_t. Let $\{X_t\}$ be a time-homogeneous It$\hat{\mathbf{o}}$ diffusion in $\mathbb{R}^d$. The infinitesimal generator A of X_t is defined by

$$Af(x) = \lim_{t\to 0} \frac{E^x[f(X_t)] - f(x)}{t} \tag{17}$$

where, $x \in \mathbb{R}^d$, X_t is the solution of It$\hat{\mathbf{o}}$ diffusion, $f : \mathbb{R}^d \to \mathbb{R}$ such that limit exists at x. E^x define expectation of x with respect to probability measure.

Consider the stationary case of It$\hat{\mathbf{o}}$ diffusion when $b = 0$ and $\sigma = I_d$ i.e.,

$$dX_t = dB_t \tag{18}$$

The generator of B_t is

$$Af = \frac{1}{2}\sum \frac{\partial^2 f}{\partial {x_i}^2} \tag{19}$$

where, $f = f(x_1, x_2, ..., x_d)$ twice differentiable, that is, $A = \frac{1}{2}\Delta$ where Δ is the Laplace operator.

Now we state Dynkin's Theorem [16]: Let f is a twice differentiable function. Suppose τ is the stopping time (escape time from a domain Ω, and $E^x[\tau] < \infty$) then

$$E^x[f(X_\tau)] = f(x) + E^x[\int_0^\tau Af(X_s)ds] \tag{20}$$

Now consider a Brownian motion $B = (B_1, B_2, ..., B_d)$ starting at $a = (a_1, a_2, ..., a_d) \in \mathbb{R}^d$ and assume $|a| < R$. We define the first exit time as τ_k of B from the ball

$$K_R = \{x \in \mathbb{R}^d; |x| < R\} \tag{21}$$

According to the Dynkin's theorem we choose $f(x) = |x|^2$ for $|x| \le R$, $X = B$ and $\tau = \sigma_k = \min(k, \tau_K)$, where k is any integer. So,

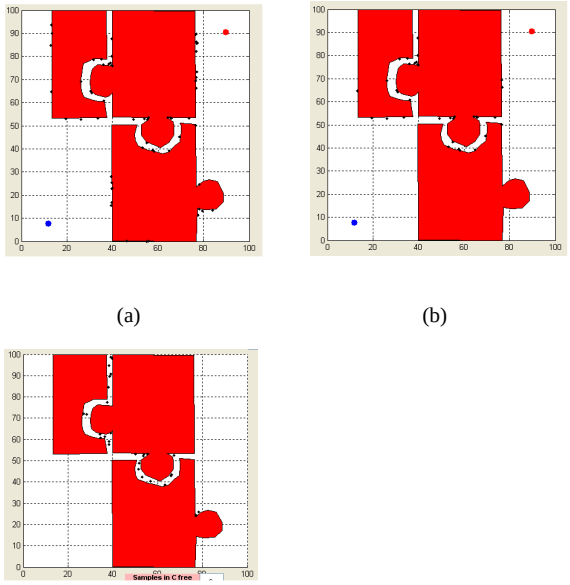
$$E^a[f(B_{\sigma_k})] = f(a) + E^a[\int_0^{\sigma_k} \frac{1}{2}\Delta f(B_s)ds] \tag{22}$$

$$= |a|^2 + E^a[\int_0^{\sigma_k} d \cdot ds] \tag{23}$$

$$= |a|^2 + d \cdot E^a[\sigma_k] \tag{24}$$

Letting $k \to \infty$, and $\tau_k = \lim \sigma_k < \infty$ a. s. and

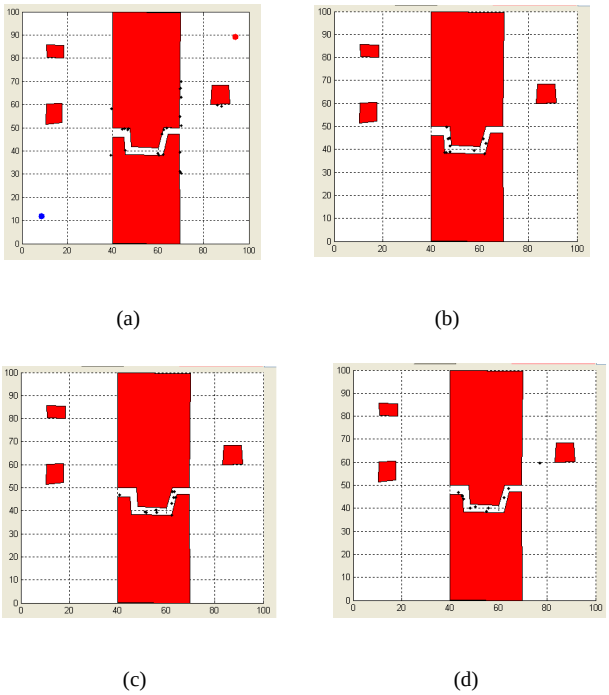
$$E^a[\tau_k] = \frac{1}{d}(R^2 - |a|^2) \tag{25}$$



(a) (b)

(c)

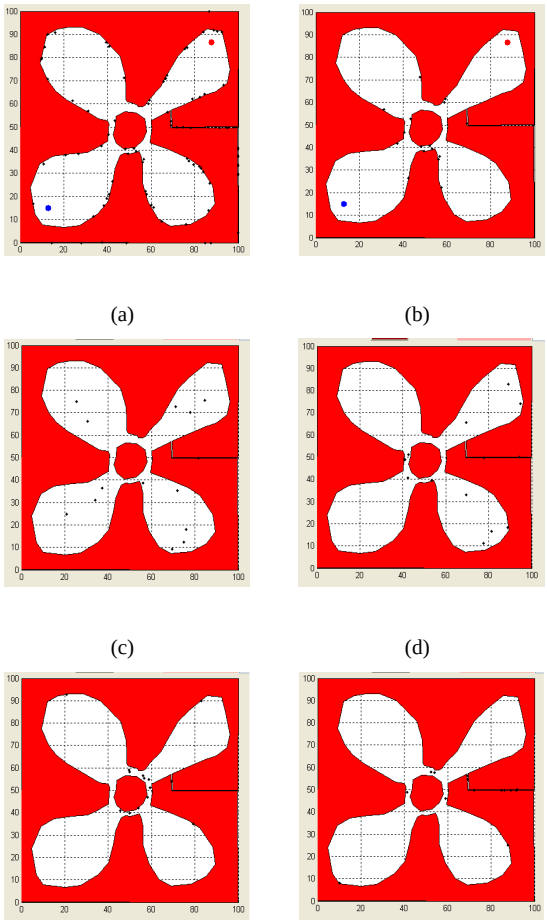
Figure 3: Problem Instance A (a) RWS without elimination (b) RWS with elimination (c) RBB (σ = 2)



(a) (b)

(c) (d)

Figure 4: Problem Instance B (a) RWS without elimination (b) RWS with elimination (c) RBB (σ = 2) (d) RBB (σ = 5)

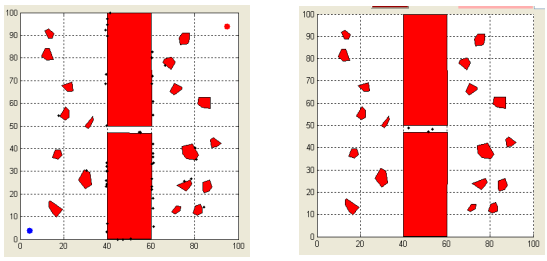


(a) (b)

(c) (d)

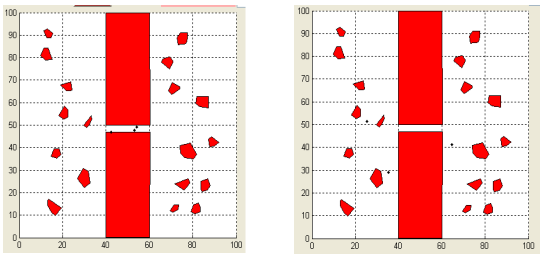
(e) (f)

Figure 5: Problem Instance C (a) RWS without elimination (b) RWS with elimination (c) RBB (σ = 20) (d) RBB (σ = 10) (e) RBB (σ = 5) (f) RBB (σ = 2)



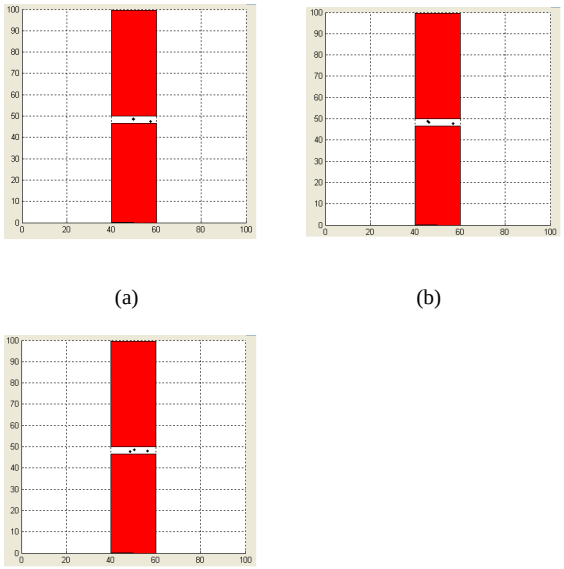
(a) (b)

Figure 6: Problem Instance D (a) RWS without elimination (b) RBB (σ = 2)



(a) (b)

Figure 7: Problem Instance D (a) RBB ($\sigma = 2$) (b) RBB ($\sigma = 10$)



(a) (b)

(c)

Figure 8: Problem Instance E (a) RWS without elimination (b) RBB ($\sigma = 2$) (c) RBB ($\sigma = 5$)

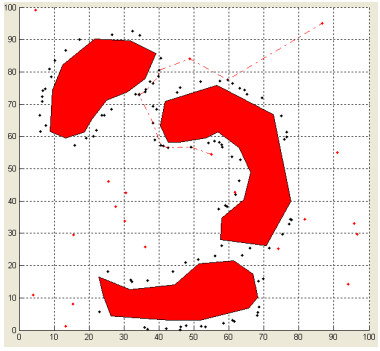


Figure 9: Problem instance A: $N_{grid} = 20$, $N_{\eta} = 100$, 3000 samples

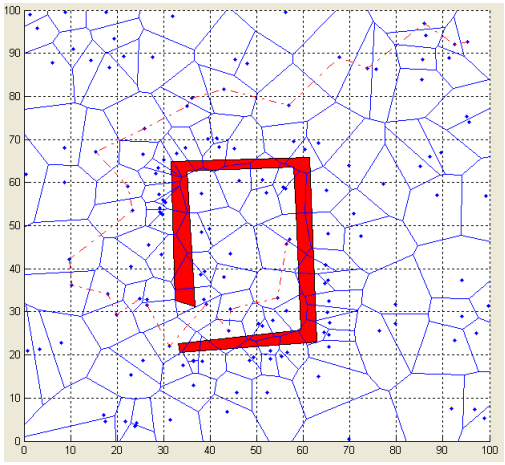


Figure 10: Problem instance B: $N_{grid} = 100$, $N_{\eta} = 30$, 5000 samples

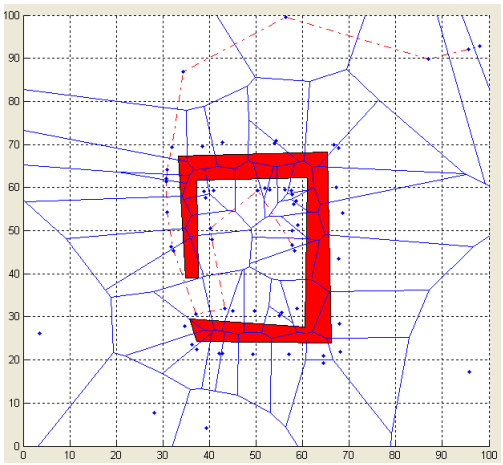


Figure 11: Problem instance C: $N_{grid} = 10$, $N_{\eta} = 50$, 3000 samples

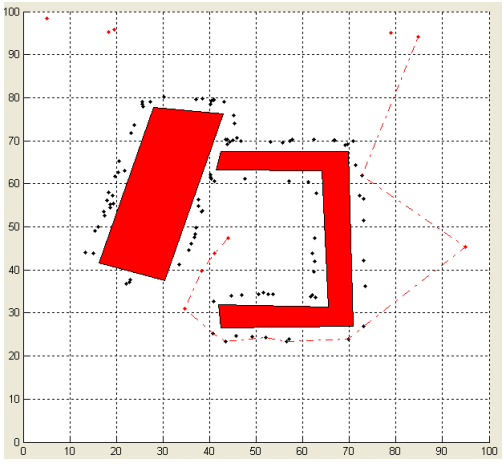


Figure 12: Problem instance D: $N_{grid} = 10$, $N_{\eta} = 100$, 5000 samples

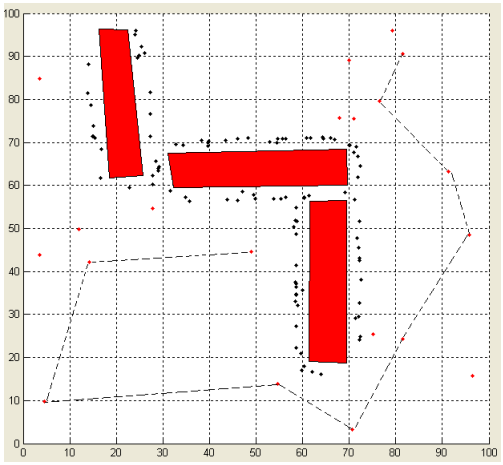


Figure 13: Problem instance E: $N_{grid} = 10$, $N_\eta = 100$, 3000 samples

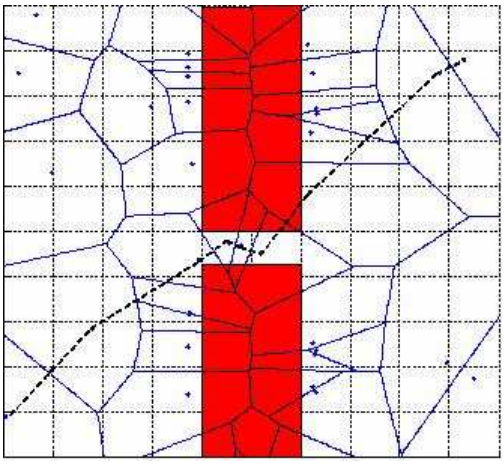


Figure 14: Problem instance F: $N_{grid} = 10$, $N_\eta = 30$, 3000 samples

Titas Bera received the B.E. degree in electrical engineering from the Vidyasagar University, WestBengal, India, in 2003, the M.E. degree in control system engineering from Jadavpur University, Kolkata, India, in 2006. Currently, he is pursuing his Ph.D. in aerospace engineering at IISc. His research interests are in the areas of path planning and control of UAVs.

M. Seetharama Bhat is a Professor in the Department of Aerospace Engineering at the Indian Institute of Science, Bangalore, India. He received the B.Sc. degree from the Mysore University, Karnataka, India, in 1972, B.E., M.E., and Ph.D. degree, from the Indian Institute of Science, Bangalore, in 1975, 1977, and 1982, respectively. His research interests are in guidance and control of aerospace vehicles, dynamics and control of smart structures and estimation techniques. He worked as a guest scientist at DLR, Germany during the year 1987. He is a fellow of the Indian National Academy of Engineering, New Delhi.

D. Ghose is a Professor in the Department of Aerospace Engineering at the Indian Institute of Science, Bangalore, India. He obtained a B.Sc. (Engg.) degree from the National Institute of Technology (formerly the Regional Engineering College), Rourkela, India, in 1982, and an M.E. and a Ph.D. degree, from Indian Institute of Science, Bangalore, in 1984 and 1990, respectively. His research interests are in guidance and control of aerospace vehicles, collective robotics and multiple agent decision-making.